

BIER
Internet-Draft
Intended status: Experimental
Expires: 13 August 2022

T. Eckert
Futurewei Technologies USA
B. Xu
Huawei Technologies (2012Lab)
9 February 2022

Carrier Grade Minimalist Multicast (CGM2) using Bit Index Explicit
Replication (BIER) with Recursive BitString Structure (RBS) Addresses
draft-eckert-bier-cgm2-rbs-01

Abstract

This memo introduces the architecture of a multicast architecture derived from BIER-TE, which this memo calls Carrier Grade Minimalist Multicast (CGM2). It reduces limitations and complexities of BIER-TE by replacing the representation of the in-packet-header delivery tree of packets through a "flat" BitString of adjacencies with a hierarchical structure of BFR-local BitStrings called the Recursive BitString Structure (RBS) Address.

Benefits of CGM2 with RBS addresses include smaller/fewer BIFT in BFR, less complexity for the network architect and in the CGM2 controller (compared to a BIER-TE controller) and fewer packet copies to reach a larger set of BFER.

The additional cost of forwarding with RBS addresses is a slightly more complex processing of the RBS address in BFR compared to a flat BitString and the novel per-hop rewrite of the RBS address as opposed to bit-reset rewrite in BIER/BIER-TE.

CGM2 can support the traditional deployment model of BIER/BIER-TE with the BIER/BIER-TE domain terminating at service provider PE routers as BFIR/BFER, but it is also the intention of this document to expand CGM2 domains all the way into hosts, and therefore eliminating the need for an IP Multicast flow overlay, further reducing the complexity of Multicast services using CGM2. Note that this is not fully detailed in this version of the document.

This document does not specify an encapsulation for CGM2/RBS addresses. It could use existing encapsulations such as [[RFC8296](#)], but also other encapsulations such as IPv6 extension headers.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Draft

bier-cgm2-rbs

February 2022

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 13 August 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Revised BSD License.

Table of Contents

1.	Overview	3
1.1.	Introduction	3
1.2.	Encapsulation Considerations	4
2.	CGM2/RBS Architecture	5
3.	CGM2/RBS forwarding plane	6
3.1.	RBS BIFT	7
3.2.	Reference encoding of RBS addresses	8
3.3.	RBS Address	8
3.3.1.	RecursiveUnit	8
3.3.2.	AddressingField	9
4.	BIER-RBS Example	9
4.1.	BFR B	10
4.2.	BFR R	12
4.3.	BFR S	13
4.4.	BFR C	14

4.5.	BFR D	14
4.6.	BFR E	15
5.	RBS forwarding Pseudocode	16
6.	Operational and design considerations (informational)	18
6.1.	Comparison with BIER-TE / BIER	18

6.1.1.	Eliminating the need for large BIFT	18
6.1.2.	Reducing number of duplicate packet copies across BFR	19
6.1.3.	BIER-TE forwarding plane complexities	20
6.1.4.	BIER-TE controller complexities	20
6.1.5.	BIER-TE specification complexities	20
6.1.6.	Forwarding plane complexity	21
6.2.	CGM2 / RBS controller considerations	21
6.3.	Analysis of performance gain with CGM2	21
6.3.1.	Reference topology	21
6.3.2.	Comparison BIER and CGM2/RBS	23
6.4.	Example use case scenarios	24
7.	Acknowledgements	24
8.	Security considerations	24
9.	Changelog	24
10.	References	24
10.1.	Normative References	24
10.2.	Informative References	25
	Authors' Addresses	25

[1.](#) Overview

[1.1.](#) Introduction

Carrier Grade Minimalist Multicast (CGM2) is an architecture derived from the BIER-TE architecture [[I-D.ietf-bier-te-arch](#)] with the following changes/improvements.

CGM2 forwarding is based on the principles of BIER-TE forwarding: It is based on an explicit, in-packet, "source routed" tree indicated through bits for each adjacency that the packet has to traverse. Like in BIER-TE, adjacencies can be L2 to a subnet local neighbor in support of "native" deployment of CGM2 and/or L3, so-called "routed" adjacencies to support incremental or partial deployment of CGM2 as needed.

The address used to replicate packets in the network is not a flat network wide BitString as in BIER-TE, but a hierarchical structure of BitStrings called a Recursive BitString Structure (RBS) Address. The significance of the BitPositions (BP) in each BitString is only local to the BIFT of the router/BFR that is processing this specific BitString.

RBS addressing allows for a more compact representation of a large set of adjacencies especially in the common case of sparse set of receivers in large Service Provider Networks (SP).

CGM2 thereby eliminates the challenges in BIER [[RFC8279](#)] and BIER-TE having to send multiple copies of the same packet in large SP networks and the complexities especially for BIER-TE (but also BIER) to engineer multiple set identifier (SI) and/or sub-domains (SD) BIER-TE topologies for limited size BitStrings (e.g.: 265) to cover large network topologies.

Like BIER-TE, CGM2 is intended to leverage a Controller to minimize the control plane complexity in the network to only a simple unicast routing underlay required only for routed adjacencies.

The controller centric architecture provides most easily any type of required traffic optimization for its multicast traffic due to their need to perform often NP-complete calculations across the whole topology: reservation of bandwidth to support CIR/PIR traffic buffer/latency to support Deterministic Network (DetNet) traffic, cost optimized Steiner trees, failure point disjoint trees for higher resilience including DetNet deterministic services.

CGM2 can be deployed as BIER/BIER-TE are specified today, by encapsulating IP Multicast traffic at Provider Edge (PE) routers, but it is also considered to be highly desirable to extend CGM2 all the way into Multicast Sender/Receivers to eliminate the overhead of an Overlay Control plane for that (legacy) IP Multicast layer and the need to deal with yet another IP multicast group addressing space. In this deployment option Controller signaling extends directly (or indirectly via BFIR) into senders.

[1.2.](#) Encapsulation Considerations

This document does not define a specific BIER-RBS encapsulation nor does it preclude that multiple different encapsulations may be beneficial to better support different use-cases or operator/user technology preferences. Instead, it discusses considerations for specific choices.

BIER-RBS can easily re-use [\[RFC8296\]](#) encapsulation. The RBS address is inserted into the [\[RFC8296\]](#) BitString field. The BFR forwarding plane needs to be configured (from Controller or control plane) that the BIFT-id(s) used with RBS addresses are mapped to BIFT and forwarding rules with RBS semantic.

SI/SD fields of [\[RFC8296\]](#) may be used as in BIER-TE, but given that CGM2 is designed (as described in the Overview section) to simplify multicast services, a likely and desirable configuration would be to only use a single BIFT in each BFR for RBS addresses, and mapping these to a single SD and SI 0.

IP Multicast [\[RFC1112\]](#) was defined as an extension of IP [\[RFC791\]](#), reusing the same network header, and IPv6 multicast inherits the same approach. In comparison, [\[RFC8296\]](#) defines BIER encapsulation as a completely separate (from IP) layer 3 protocol, and duplicates both IP and MPLS header elements into the [\[RFC8296\]](#) header. This not only results in always unused, duplicate header parameters (such as TC vs. DSCP), but it also foregoes the option to use any non-considered IPv6 extension headers with BIER and would require the introduction of a whole new BIER specific socket API into host operating systems if it was to be supported natively in hosts.

Therefore an encapsulation of RBS addresses using an IP and/or IPv6 extension header may be more desirable in otherwise IP and/or IPv6 only deployments, for example when CGM2 is extended into hosts, because it would allow to support CGM2 via existing IP/IPv6 socket APIs as long as they support extension headers, which the most important host stacks do today.

[2.](#) CGM2/RBS Architecture

This section describes the basic CGM2 architecture via Figure 1 through its key differences over the BIER-TE architecture.

(instead of the BIER-TE (BitString,SI,SD)) to be imposed as part of the RBS address header (compared to the BIER encapsulation [[RFC8296](#)]) on the BFIR.

To eliminate the need for an IP Multicast flow overlays, a CGM2 domain may extend all the way into Sender/Receiver hosts. This is called "end-to-end" deployment model. In that case, the sender host and CGM2 controller collaborate to determine the desired receivers for a packet as well as desired path policy/requirements, the controller indicates to the sender of the packet the necessary RBS address and address of the BFIR, and the Sender imposes an appropriate RBS address header together with a unicast encapsulation towards the BFIR.

CGM2 is also intended so especially simplify controller operations that also instantiate QoS policies for multicast traffic flows, such as bandwidth and latency reservations (e.g.: DetNet). As in BIER-TE, this is orthogonal to the operations of the CGM2/RBS address forwarding operations and will be covered in separate documents.

[3.](#) CGM2/RBS forwarding plane

Instead of a (flat) BitString as in BIER-TE that use a network wide shared BP address space for adjacencies across multiple BFR, CGM2 uses a structured address built from so-called RecursiveUnits (RU) that contain BitStrings, each of which is to be parsed by exactly one BFR along the delivery tree of the packet.

The equivalent to a BIER/BIER-TE BitString is therefore called the RecursiveUnit BitString Structure (RBS) Address. Forwarding for CGM2 is therefore also called RBS forwarding.

[3.1.](#) RBS BIFT

RBS BIFT as shown in Figure 2 are, like BIER-TE BIFT, tables that are indexed by BP, containing for each BP an adjacency. The core difference over BIER-TE BIFT is that the BP of the BIFT are all local to the BFR, whereas in BIER-TE, the BP are shared across a BIER-TE domain, each BFR can only use a subset the BP for its own adjacencies, and only in some cases can BP be shared for adjacencies

across two (or more) BFR. Because of this difference, most of the complexities of BIER-TE BIFT are not required with BIER-RBS BIFT, see [Section 6.1.3](#).

BP		Recursive	Adjacency
1		1	adjacent BFR
2		0	punt/host
	
N

Figure 2: RBS BIFT

An RBS BIFT has a configured number of N addressable BP entries. When a BFR receives a packet with an RBS address, it expects that the BitString inside the RBS address that needs to be parsed by the BFR (see [Section 3.3](#) has a length that matches N according to the encapsulation used for the RBS address. Therefore, N MUST support configuration in increments of the supported size of the BitString in the encapsulation of the RBS Address. In the reference encoding (see [Section 3.3](#)), the increment for N is 1 (bit). If an encapsulation would call for a byte accurate encoding of the BitString, N would have to be configurable in increments of 8.

BFR MUST support a value of N larger than the maximum number of adjacencies through which RBS forwarding/replication of a single packet is required, such as the number of physical interfaces on BFR that are intended to be deployed as a Provider Core (P) routers.

RBS BIFT introduce a new "Recursive" flag for each BP. These are used for adjacencies to other BFR to indicate that the BFR processing the packet RBS address BitString also has to expect for every BP with the recursive flag set another RU inside the RBS address.

Structure elements of the RBS Address and its components are parameterized according to a specific encapsulation for RBS addresses, such as the total size of the TotalLen field and the unit in which it is counted (see [Section 3.3](#)). These parameters are outside the scope of this document. Instead, this document defines example parameters that together form the so called "Reference encoding of RBS addresses". This encoding may or may not be adopted for any particular encapsulation of RBS addresses.

[3.3](#). RBS Address

An RBS address is structured as shown in Figure 3.

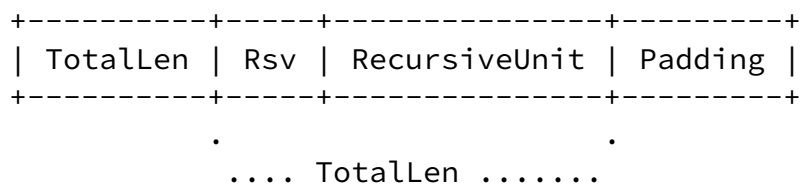


Figure 3: RBS Address

TotalLen counts in some unit, such as bits, nibbles or bytes the length of the RBS Address excluding itself and Padding. For the reference encoding, TotalLen is an 8-bit field that counts the size of the RBS address in bits, permitting for up to 256 bit long RBS addresses.

In case additional, non-recursive flags/fields are determined to be required in the RBS Address, they should be encoded in a field between TotalLen and RecursiveUnit, which is called Rsv. In the reference encoding, this field has a length of 0.

Padding is used to align the RBS address as required by the encapsulation. In the reference encoding, this alignment is to 8 bits (byte boundaries). Therefore, Padding (bits) = (8 - TotalLen % 8).

[3.3.1](#). RecursiveUnit

The RecursiveUnit field is structured as shown in Figure 4.

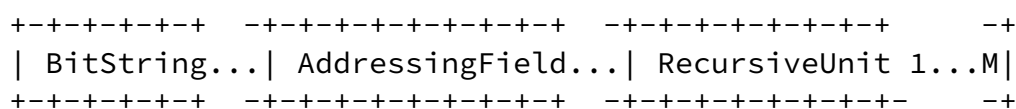


Figure 4: RBS RecursiveUnit

The BitString field indicates the bit positions (BPs) to which the packet is to be replicated using the BIFT of the BFR that is processing the Recursive unit.

For each of M BP set in the BitString of the RecursiveUnit for which the Recursive flag is set in the BIFT of the BFR, the RecursiveUnit contains a RecursiveUnit i , $i=1\dots M$, in order of increasing BP index.

If adjacencies between BFR are not configured as recursive in the BIFT, this recursive extraction does not happen for an adjacency, no RecursiveUnit i has to be encoded for the BP, and BFRs across such adjacencies would have to share the BP of a common BIFT as in BIER-TE. This option is not further discussed in this version of the document.

3.3.2. AddressingField

The AddressingField of an RBS address is structured as shown in Figure 5.

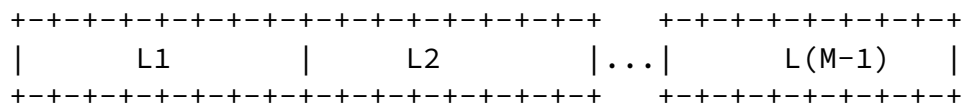


Figure 5: RBS AddressingField

The AddressingField consists of one or more fields L_i , $i=1\dots(M-1)$. L_i is the length of RecursiveUnit i for the i 'th recursive bit set in the BitString preceding it.

In the reference encoding, the lengths are 8-bit fields indicating the length of RecursiveUnits in bits.

The length of the M 'th RecursiveUnit is not explicitly encoded but has to be calculated from TotalLen.

4. BIER-RBS Example

Figure 6 shows an example for RBS forwarding.

Internet-Draft

bier-cgm2-rbs

February 2022

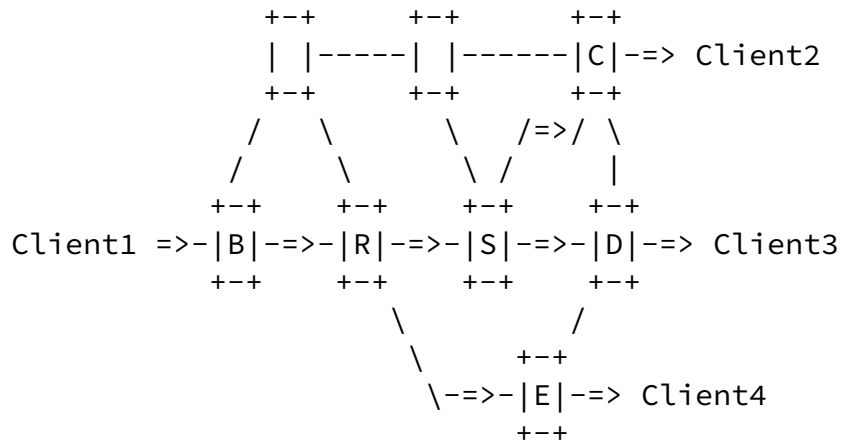


Figure 6: Example Network Topology

A packet from Client1 connected to BFIR B is intended to be replicated to Client2,3,4. The example initially assumes the traditional option of the architecture, in which the imposition of the header for the RBS address happens on BFIR B, for example based on functions of an IP multicast flow overlay.

A controller determines that the packet should be forwarded hop-by-hop across the network as shown in Figure 7.

```

Client 1 ->B(impose BIER-RBS)
=>R(
=> E (dispose BIER-RBS)
=> Client4
=> S(
=>C (dispose BIER-RBS)
=> Client2
=>D (dispose BIER-RBS)
=> Client3
)
)
  
```

Figure 7: Desired example forwarding tree

[4.1.](#) BFR B

The 34 bit long (without padding) RBS address shown in Figure 8 is constructed to represent the desired tree from Figure 7 and is imposed at B onto the packet through an appropriate header supporting the reference encoding of RBS addresses.

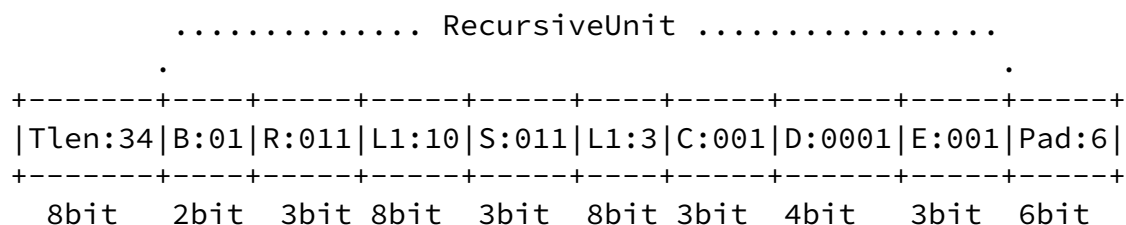


Figure 8: RBS Address imposed at BFIR-B

In Figure 8 and further the illustrations of RBS addresses, BitStrings are preceded by the name of the BFR for whom they are destined and their values are shown as binary with the lowest BP 1 starting on the left. TotalLength (Tlen:), AddressingField (L1:) and Padding (Pad:) fields are shown with decimal values.

RBS forwarding on B examines this address based on its RBS BIFT with N=2 BP entries, which is shown in Figure 9.

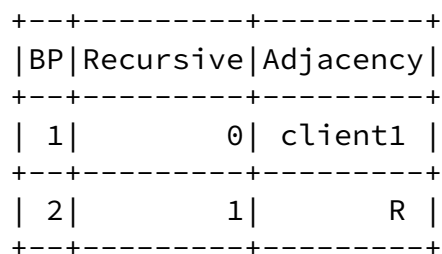


Figure 9: BIER-RBS BIFT on B

This results in the parsing of the RBS address as shown in Figure 10, which shows that B does not need (nor can) parse all structural elements, but only those relevant to its own RBS forwarding procedure.

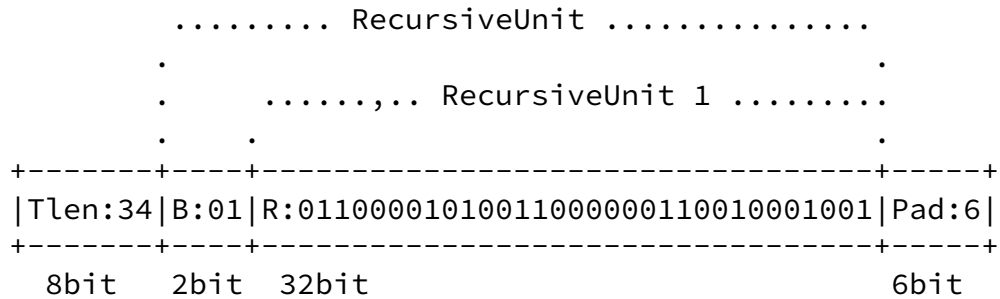


Figure 10: RBS Address as processed by BFIR-B

There is only one BP towards BFR R set in the BitString B:01, so the RecursiveUnit 1 follows directly after the end of the BitString B:01 and it covers the whole Tlen - length of BitString (34 - 2 = 32 bit).

B rewrites the RBS address by replacing the RecursiveUnit with RecursiveUnit 1 and adjusts the Padding to zero bits. The resulting RBS address is shown in Figure 11. It then sends the packet copy with that rewritten RBS address to BFR R.

[4.2.](#) BFR R

BFR R receives from BFR B the packet with that RBS address shown in Figure 11.



Figure 11: RBS Address processed by BFR-R

BFR R parses the RBS Address as shown in Figure 12 using its RBS BIFT of N=3 BP entries shown in Figure 13.

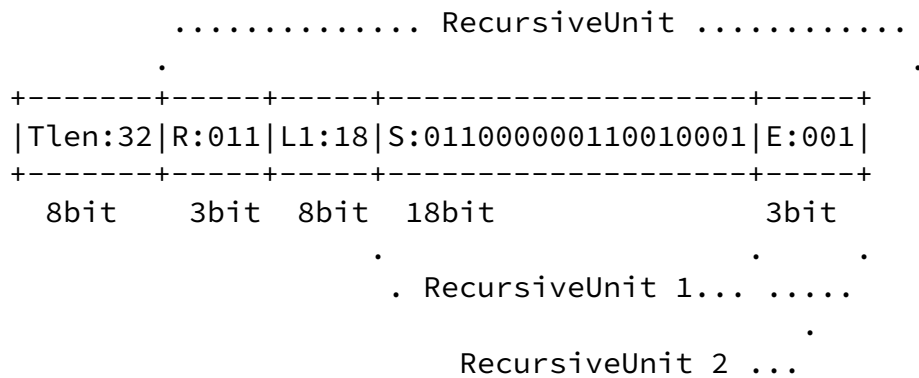


Figure 12: RBS Address processed by BFR-R

Because there are two recursive BP set in the BitString for R, one for BFR S and one for BFR E, one Length field L1 is required in the AddressingField, indicating the length of the RecursiveUnit 1 for BFR S, followed by the remainder of the RBS address being the RecursiveUnit 2 for BFR E.

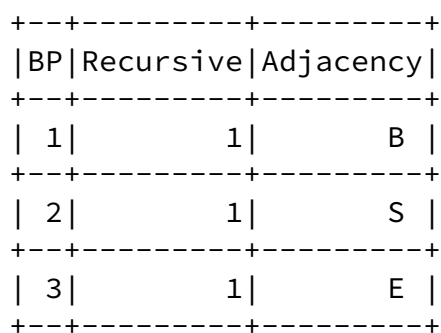


Figure 13: RBS BIFT on BFR R

BFR R accordingly creates one copy for BFR S using RecursiveUnit 1, and only copy for BFR E using RecursiveUnit 2, updating Padding accordingly for each copy.

[4.3.](#) BFR S

BFR S receives from BFR B the packet and parses the RBS address as shown in Figure 14 using its RBS BIFT of N=3 BP shown in Figure 15.

```

      .... RecursiveUnit ....
      .
+-----+-----+-----+-----+-----+
|Tlen:18|S:011|L1:3|C:001|D:0001|Pad:6|
+-----+-----+-----+-----+-----+
  8bit   3bit 8bit  3bit  4bit  3bit
      . . . . .
      .... ..
RecursiveUnit 1 .
RecursiveUnit 2 .....

```

Figure 14: RBS Address processed by BFR-S

```

+---+-----+-----+
|BP|Recursive|Adjacency|
+---+-----+-----+
| 1|          1|        R |
+---+-----+-----+
| 2|          1|        C |
+---+-----+-----+
| 3|          1|        D |
+---+-----+-----+

```

Figure 15: RBS BIFT on BFR-S

BFR S accordingly sends one packet copy with RecursiveUnit 1 in the RBS address to BFR C and a second packet copy with RecursiveUnit 2 to BFR D.

[4.4.](#) BFR C

BFR C receives from BFR S the packet and parses the RBS address according to its N=3 BP entries BIFT (shown in Figure 17) as shown in Figure 16.

```

+-----+-----+-----+

```

```

|Tlen:3 |C:001|Pad:5|
+-----+-----+-----+
      8bit      3bit 5bi

```

Figure 16: RBS Address processed by BFR-C

```

+---+-----+-----+-----+
|BP|Recursive|Adjacency|
+---+-----+-----+-----+
| 1|          1|          S |
+---+-----+-----+-----+
| 2|          1|          D |
+---+-----+-----+-----+
| 3|          0| local_decap|
+---+-----+-----+-----+

```

Figure 17: RBS BIFT on BFR-C

BFR S accordingly creates one packet copy for BP 3 where the RBS address encapsulation is disposed of, and the packet is ultimately forwarded to Client 2, for example because of an IP multicast payload for which the multicast flow overlay identifies Client 2 as an interested receiver, as in BIER/BIER-TE.

To avoid having to use an IP flow overlay, the BIFT could instead have one BP allocated for every non-RBS destination, in this example BP 3 would then explicitly be allocated for Client 2, and instead of disposing of the RBS address encapsulation, BFR C would impose or rewrite a unicast encapsulation to make the packet become a unicast packet directed to Client 2. This option is not further detailed in this version of the document.

[4.5.](#) BFR D

The procedures for processing of the packet on BFR D are very much the same as on BFR C. Figure 18 shows the RBS address at BFR D, Figure 19 shows the N=4 bit RBS BIFT of BFR D.

```

+-----+-----+-----+
|Tlen:4 |D:0001|Pad:4|
+-----+-----+-----+
      8bit      4bit      4bit

```


Figure 18: RBS Address processed by BFR-D

BP	Recursive	Adjacency
1	1	S
2	1	C
3	1	E
4	0	local_decap

Figure 19: RBS BIFT on BFR-D

4.6. BFR E

The procedures for processing of the packet on BFR E are very much the same as on BFR C and D. Figure 20 shows the RBS address at BFR D, Figure 21 shows the N=E bit RBS BIFT of BFR E.

Tlen:3	E:001	Pad:5
8bit	3bit	5bit

Figure 20: RBS Address processed by BFR-E

BP	Recursive	Adjacency
1	1	R
2	1	D
3	0	local_decap

Figure 21: RBS BIFT on BFR-E

[5.](#) RBS forwarding Pseudocode

The following example RBS forwarding Pseudocode assumes the reference encoding of bit-accurate length of BitStrings and RecursiveUnits as well as 8-bit long TotalLen and AddressingField Lengths. All packet field addressing and address/offset calculations is therefore bit-accurate instead of byte accurate (which is what most CPU memory access today is).

Internet-Draft

bier-cgm2-rbs

February 2022

```
void ForwardRBSPacket (Packet)
{
    RBS = GetPacketMulticastAddr(Packet);
    Total_len = RBS;
    Rsv = Total_len + length(Total_Len);
    BitStringA = Rsv + length(Rsv);
    AddressingField = BitStringA + BIFT.entries;

    // [1] calculate number of recursive bits set in BitString
    CopyBitString(*BitStringA, *RecursiveBits, BIFT.entries);
    And(*RecursiveBits,*BIFTRecursiveBits, BIFT.entries);
    N = CountBits(*RecursiveBits, BIFT.entries);

    // Start of first RecursiveUnit in RBS address
    // After AddressingField array with 8-bit length fields
    RecursiveUnit = AddressingField + (N - 1) * 8;

    RemainLength = *Total_len - length(Rsv)
                  - BIFT.entries;

    Index = GetFirstBitPosition(*BitStringA);
    while (Index) {
        PacketCopy = Copy(Packet);

        if (BIFT.BP[Index].recursive) {
            if(N == 1) {
                RecursiveUnitLength = RemainLength;
            } else {
                RecursiveUnitLength = *AddressingField;
                N--;
                AddressingField += 8;
                RemainLength -= RecursiveUnitLength;
                RemainLength -= 8; // 8 bit of AddressingField
            }
            RewriteRBS(PacketCopy, RecursiveUnit, RecursiveUnitLength);
            SendTo(PacketCopy, BIFT.BP[Index].adjacency);

            RecursiveUnit += RecursiveUnitLength;
        } else {
            DisposeRBSheader(PacketCopy);
        }
    }
}
```

```

        SendTo(PacketCopy, BIFT.BP[Index].adjacency);
    }
    Index = GetNextBitPosition(*BitStringA, Index);
}

```

Figure 22: RBS address forwarding Pseudocode

Explanations for Figure 22.

RBS is the (bit accurate) address of the RBS address in packet header memory. BitStringA is the address of the RBS address BitString in memory. length(Total_Len) and length(Rsv) are the bit length of the two RBS address fields, e.g.: 8 bit and 0 bit for the reference encoding.

The BFR local BIFT has a total number of BIFT.entries addressable BP 1...BIFTentries. The BitString therefore has BIFT.entries bits.

BIFT.RecursiveBits is a BitString pre-filled by the control plane with all the BP with the recursive flag set. This is constructed from the Recursive flag setting of the BP of the BIFT. The code starting at [1] therefore counts the number of recursive BP in the packets BitString.

Because the AddressingField does not have an entry for the last (or only) RecursiveUnit, its length has to be calculated by taking TotalLen into account.

RewriteRBS needs to replace RBS address with the RecursiveUnit address, keeping only Rsv, recalculating TotalLen and adding appropriate Padding.

For non-recursive BP, the Pseudocode assumes disposition of the RBSheader. This is not strictly necessary but non-disposing cases are outside of scope of this version of the document.

[6.](#) Operational and design considerations (informational)

[6.1.](#) Comparison with BIER-TE / BIER

This section discusses informationally, how and where CGM2 can avoid different complexities of BIER/BIER-TE, and where it introduces new

complexities.

6.1.1. Eliminating the need for large BIFT

In a BIER domain with M BFER, every BFR requires M BIFT entries. If the supported BSL is N and $M > 2^N$, then $S = (M / 2^N)$ set indices (SI) are required, and S copies of the packet have to be sent by the BFIR to reach all targeted BFER.

In CGM2, the number of BIFT entries does not need to scale with the number of BFER or paths through the network, but can be limited to only the number of L2 adjacencies of the BFR. Therefore CGM2 requires minimum state maintenance on each BFR, and multiple SI are not required.

6.1.2. Reducing number of duplicate packet copies across BFR

If the total size of an RBS encoded delivery tree is larger than a supported maximum RBS header size, then the CGM2 controller simply needs to divide the tree into multiple subtrees, each only addressing a part of the BFER (leaves) of the target tree and pruning any unnecessary branches.

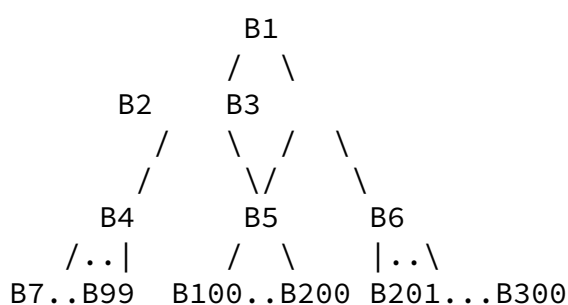


Figure 23: Simple Topology Example

Consider the simple topology in Figure 23 and a multicast packet that needs to reach all BFER B7...B300. Assume that the desired maximum RBM header size is such that a RBS address size of ≤ 256 bits is desired. The CGM2 controller could create an RBS address $B1 \Rightarrow B2 \Rightarrow B4 \Rightarrow (B7..B99)$, for a first packet, an RBS address $B1 \Rightarrow B3 \Rightarrow B5 \Rightarrow (B100..B200)$ for a second packet and a third RBS address $B1 \Rightarrow B3 \Rightarrow B6 \Rightarrow B201...B300$.

The elimination of larger BIFT state in BFR through multiple SI in BIER/BIER-TE does come at the expense of replicating initial hops of a tree in RBS addresses, such as in the example the encoding of B1=>B3 in the example.

Consider that the assignment of BFIR-ids with BIER in the above example is not carefully engineered. It is then easily possible that the BFR-ids for B7..B99 are not sequentially, but split over a larger BFIR-id space. If the same is true for all BFER, then it is possible that each of the three BFR B4,B5 and B6 has attached BFER from three different SI and one may need to send for example three multiple packets to B7 to address all BFER B7..B99 or to B5 to address all B100..B200 or B6 to address all B201...B300. These unnecessary duplicate packets across B4, B5 or B6 are because of the addressing principle in BIER and are not necessary in CGM2, as long as the total length of an RBS address does not require it.

For more analysis, see [Section 6.3](#).

[6.1.3](#). BIER-TE forwarding plane complexities

BIER-TE introduces forwarding plane complexities to allow reducing the BSL required. While all of these could be supported / implemented with CGM2, this document contends that they are not necessary, therefore providing significant overall simplifications.

- * BIER-TE supports multiple adjacencies in a single BIFT Index to allow compressing multiple adjacencies into a single Index for traffic that is known to always require replications to all those adjacencies (such as when flooding TV traffic).
- * BIER-TE support ECMP adjacencies which have to calculate which out of 2 or more possible adjacencies a packet should be forwarded to.
- * BIER-TE supports special Do-Not-Clear (DNC) behavior of adjacencies to permit reuse of such a bit for adjacencies on multiple consecutive BFR. This behavior specifically also raises the risk of looping packets.

[6.1.4.](#) BIER-TE controller complexities

BIER-TE introduces BIER-TE controller plane mechanisms that allow to reuse bits of the flat BIER-TE BitStrings across multiple BFR solely to reduce the number of BP required but without introducing additional complexities for the BIER-TE forwarding plane.

- * Shared BP for all Leaf BFR.
- * Shared BP for both Interfaces of p2p links.
- * Shared bits for multi-access subnets (LANs).

These bit-sharing mechanisms are unnecessary and inapplicable to CGM2 because there is no need to share BP across the BIFT of multiple BFR.

[6.1.5.](#) BIER-TE specification complexities

The BIER-TE specification distinguishes between forward (link scope) and routed (underlay routed) adjacencies to highlight, explain and emphasize on the ability of BIER-TE to be deployed in an overlay fashion especially also to reduce the necessary BSL, even when all routers in the domain could or do support BIER-TE.

In CGM2, routed adjacencies are considered to be only required in partial deployments to forward across non-CGM2 enabled routers. This specification does therefore not highlight link scope vs. routed adjacencies as core distinct features.

[6.1.6.](#) Forwarding plane complexity

CGM2 introduces some more processing calculation steps to extract the BitString that needs to be examined by a BFR from the RBS address. These additional steps are considered to be non-problematic for todays programmable forwarding planes such as P4.

Whereas BIER-TE clears bit on each hops processing, CGM2 rewrites the address on every hop by extracting the recursive unit for the next hop and make it become the packet copies address. This rewrite shortens the RBS address. This hopefully has only the same complexity as (tunnel) encapsulations/decapsulations in existing

forwarding planes.

[6.2.](#) CGM2 / RBS controller considerations

TBD. Any aspects not covered in [Section 6.1](#).

[6.3.](#) Analysis of performance gain with CGM2

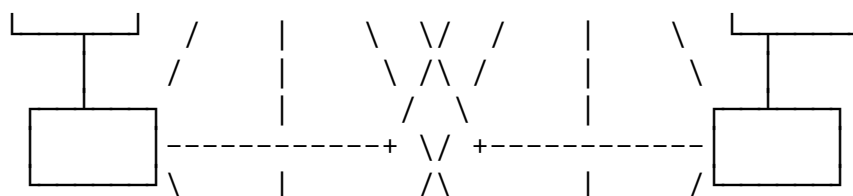
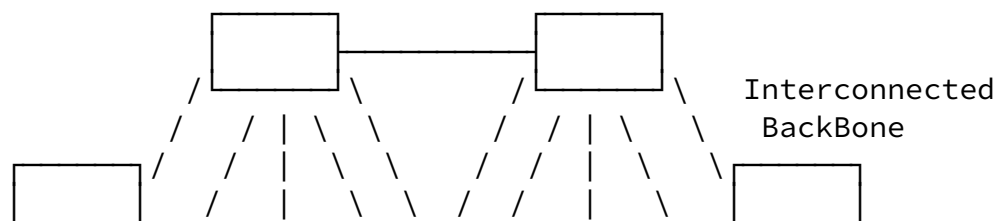
TBD: Comparison of number of packets/header sizes required in large real-world operator topology between BIER/BIER-TE and CGM2.

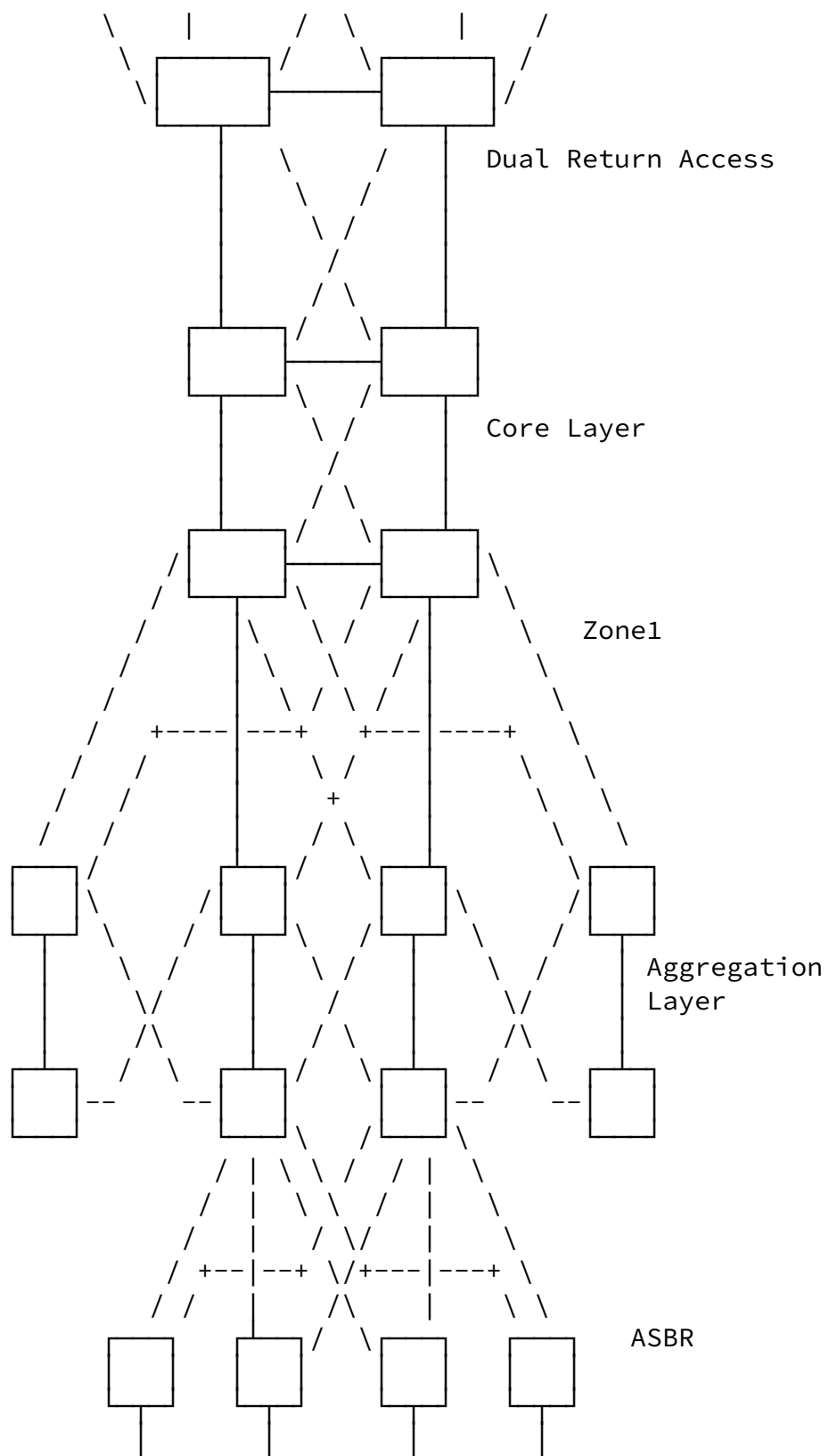
Analysis: Gain in dense topology.

[6.3.1.](#) Reference topology

Reference topology description:

1. Typical topology of Beijing Mobile in China.
2. All zones dual homing access to backbone.
3. Core layer: 4 nodes full mesh connected
4. Aggregation layer: 8 nodes are divided into two layers, with full interconnection between the layers and dual homing access to the core layer on the upper layer.
5. Aggregation rings: 8 rings, 6 nodes per ring
6. Access rings: 3600 nodes, 18 nodes per ring.





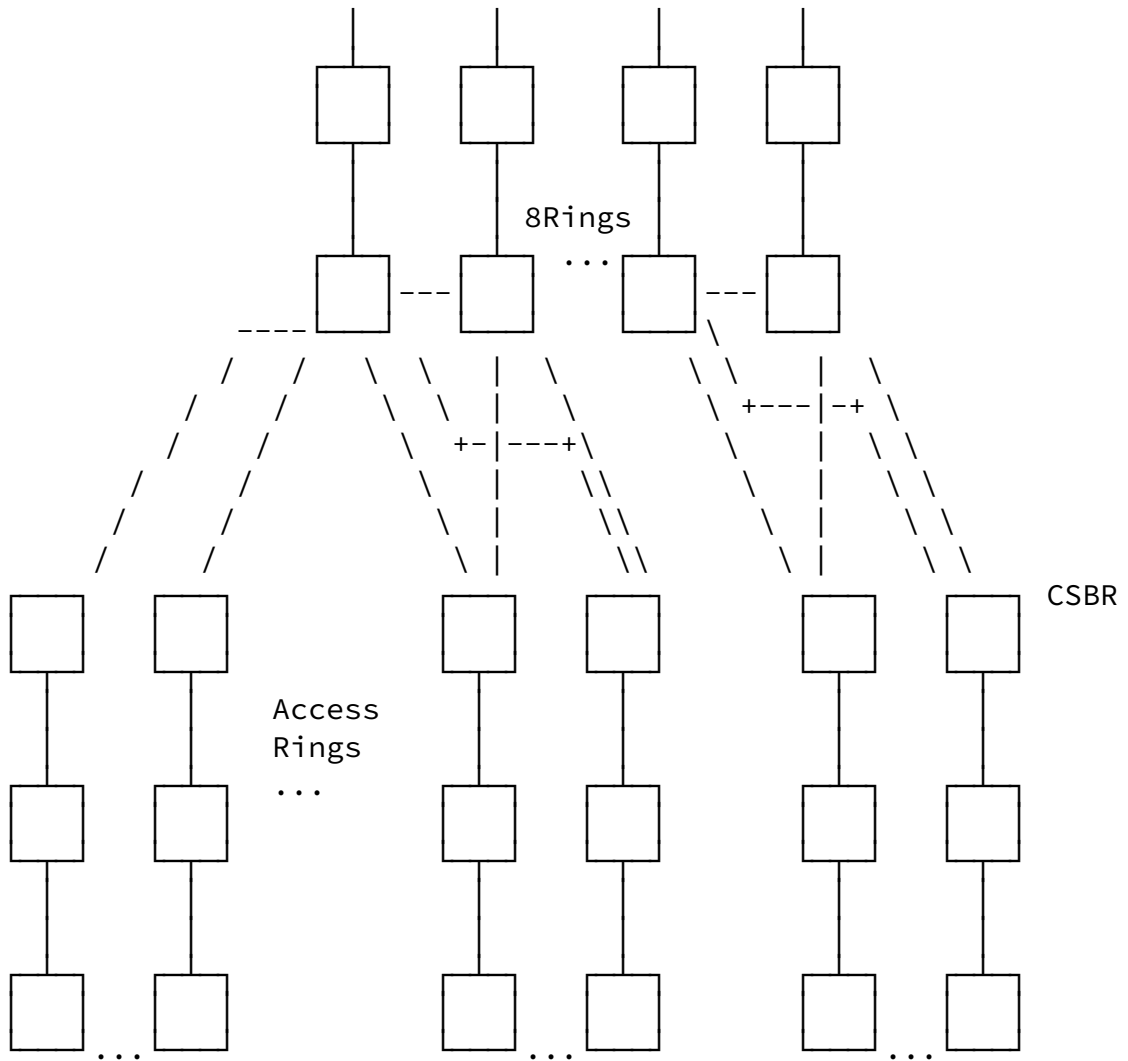


Figure 24: Reference Topology

6.3.2. Comparison BIER and CGM2/RBS

The following performance comparison is based on Figure 24.

1. CGM2: We randomly select egress points as group members, with the total number ranging from 10 to 28800 (for sake of simplicity, we assume merely one client per egress point). The egress points are randomly distributed in the topology with 10 runs for each value, showing the average result in our graphs. The total number of samples is 60
2. BIER: We divide the overall topology into 160 BIER domains, each of which includes 180 egress points, providing the total of 28800 egress points.

Internet-Draft

bier-cgm2-rbs

February 2022

3. Simulation: In order to compare the BIER against the in-packet tree encoding mechanism, we limit the size of the header to 256 bits (the typical size of a BIER header).

Conclusion: 1. BIER reaches its 160 packet replication limit at about 500 users, while the in-packet tree encoding reaching its limit of 125 replications at about 12000 users. And the following decrease of replications is caused by the use of node-local broadcast as a further optimization. 2. For the sake of comparison, the same 256-bit encapsulation limit is imposed on CGM2, but we can completely break the 256-bit encapsulation limit, thus allowing the source to send fewer multicast streams. 3. CCGM2 encoding performs significantly better than BIER in that it requires less packet replications and there network bandwidth.

6.4. Example use case scenarios

TBD.

7. Acknowledgements

This work is based on the design published by Sheng Jiang, Xu Bing, Yan Shen, Meng Rui, Wan Junjie and Wang Chuang {jiangsheng|bing.xu|yanshen|mengrui|wanjunjie2|wangchuang}@huawei.com, see [[CGM2Design](#)].

8. Security considerations

TBD.

9. Changelog

[RFC-Editor: please remove this section].

This document is written in <https://github.com/cabo/kramdown-rfc2629> markup language. This documents source is maintained at <https://github.com/toerless/bier-cgm2-rbs>, please provide feedback to the appropriate IETF mailing list and submit an Issue to the GitHub.

01 - Added [section 6.3](#) about performance comparison and co-author (Robin).

00 - Initial version from [[CGM2Design](#)].

10. References

10.1. Normative References

[I-D.ietf-bier-te-arch]

Eckert, T., Menth, M., and G. Cauchie, "Tree Engineering for Bit Index Explicit Replication (BIER-TE)", Work in Progress, Internet-Draft, [draft-ietf-bier-te-arch-12](https://www.ietf.org/archive/id/draft-ietf-bier-te-arch-12), 28 January 2022, <<https://www.ietf.org/archive/id/draft-ietf-bier-te-arch-12.txt>>.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, [RFC 1112](https://www.rfc-editor.org/info/rfc1112), DOI 10.17487/RFC1112, August 1989, <<https://www.rfc-editor.org/info/rfc1112>>.

[RFC791] Postel, J., "Internet Protocol", STD 5, [RFC 791](https://www.rfc-editor.org/info/rfc791), DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.

[RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", [RFC 8279](https://www.rfc-editor.org/info/rfc8279), DOI 10.17487/RFC8279, November 2017, <<https://www.rfc-editor.org/info/rfc8279>>.

[RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", [RFC 8296](https://www.rfc-editor.org/info/rfc8296), DOI 10.17487/RFC8296, January 2018, <<https://www.rfc-editor.org/info/rfc8296>>.

10.2. Informative References

[CGM2Design]

Jiang, S., Xu, B.(., Shen, Y., Rui, M., Junjie, W., and W. Chuang, "Novel Multicast Protocol Proposal Introduction", 10 October 2021, <<https://github.com/BingXu1112/CGMM/blob/main/Novel%20Multicast%20Protocol%20Proposal%20Introduction.pptx>>.

Authors' Addresses

Toerless Eckert
Futurewei Technologies USA
2220 Central Expressway
Santa Clara, CA 95050
United States of America

Email: tte@cs.fau.de

Eckert & Xu

Expires 13 August 2022

[Page 25]

Internet-Draft

bier-cgm2-rbs

February 2022

Bing (Robin) Xu
Huawei Technologies (2012Lab)
China

Email: bing.xu@huawei.com

